Directories

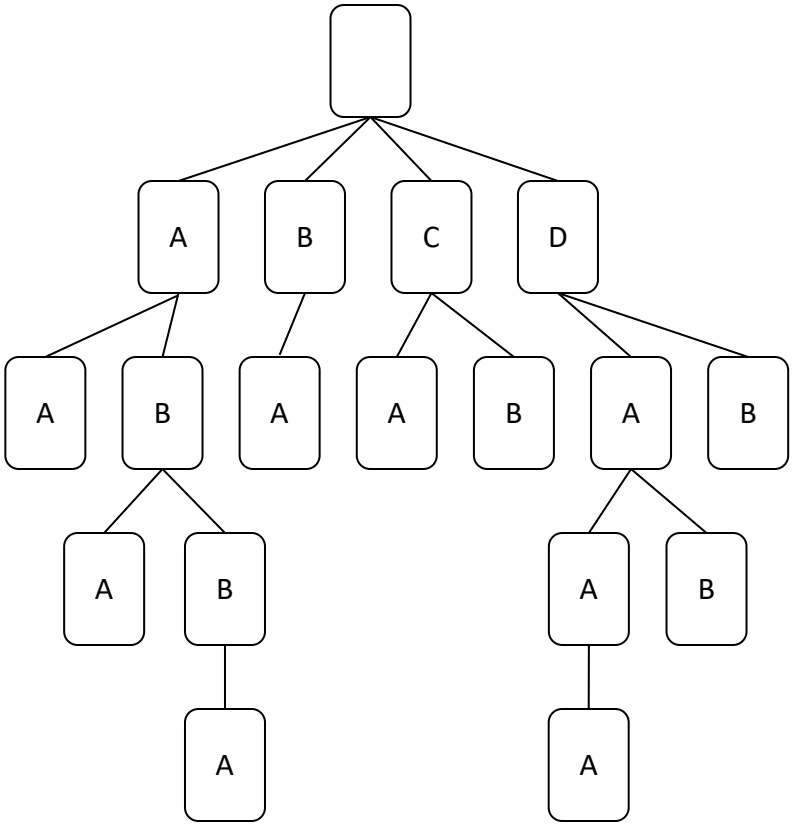
Peter J. Denning

Directory

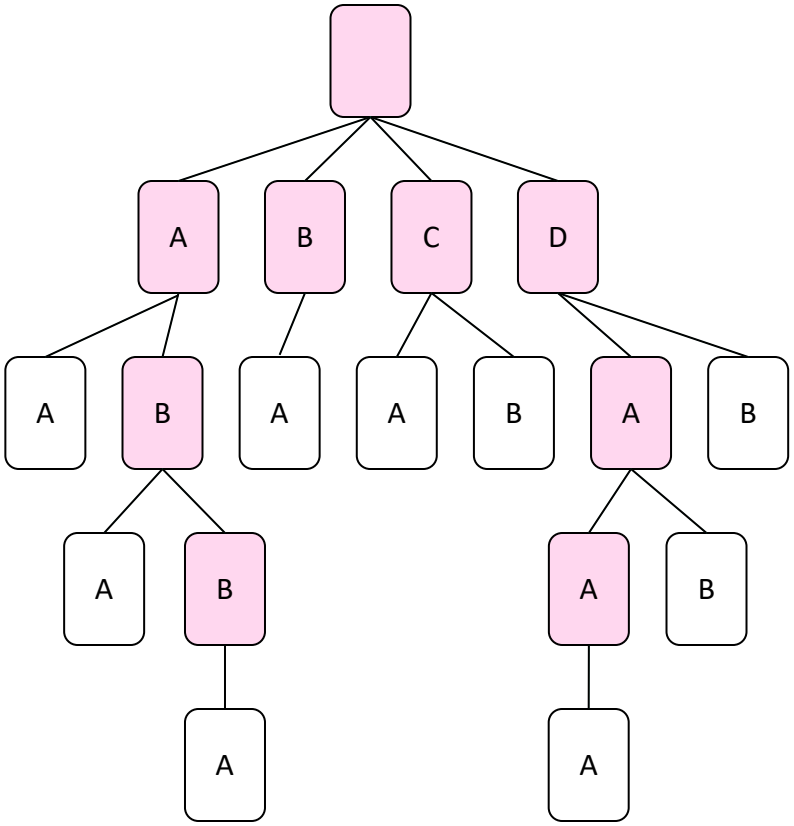
- Maps user-chosen symbolic name to capability
- Managed in user kernel
- Directories can point to other directories, stream objects, or digital object for which capabilities are defined

Directories come in trees

- Why?
  - Human desires for hierarchical organization
  - Localized management – related objects kept together
- Structure of directory trees
- Pathnames (are unique)



The hierarchy of directories and other digital objects expresses human desires for grouping similar things into hierarchies.

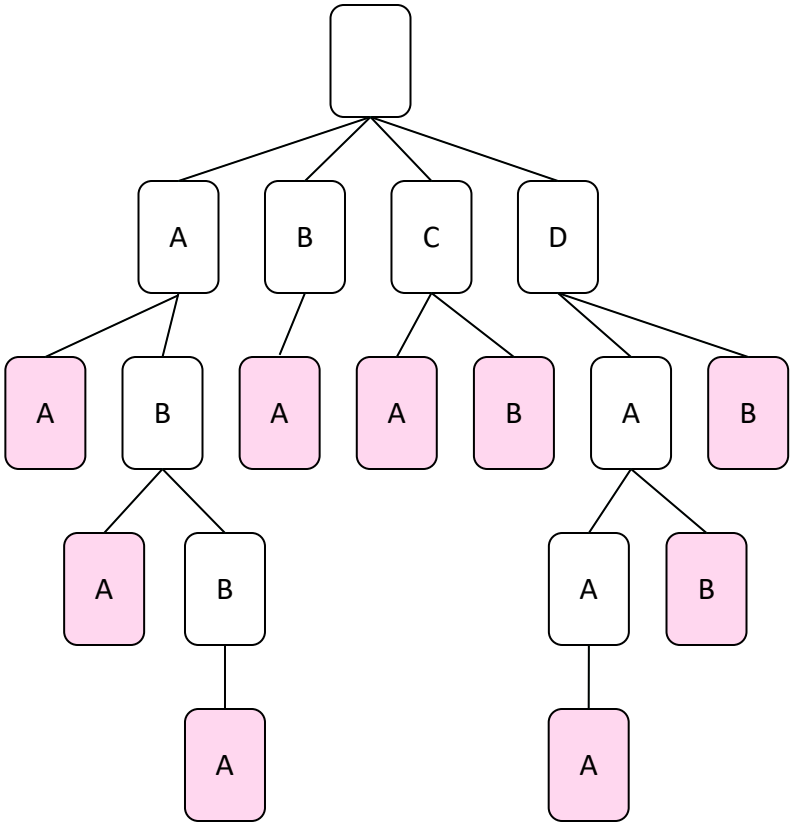


The top node of the the tree is the root.

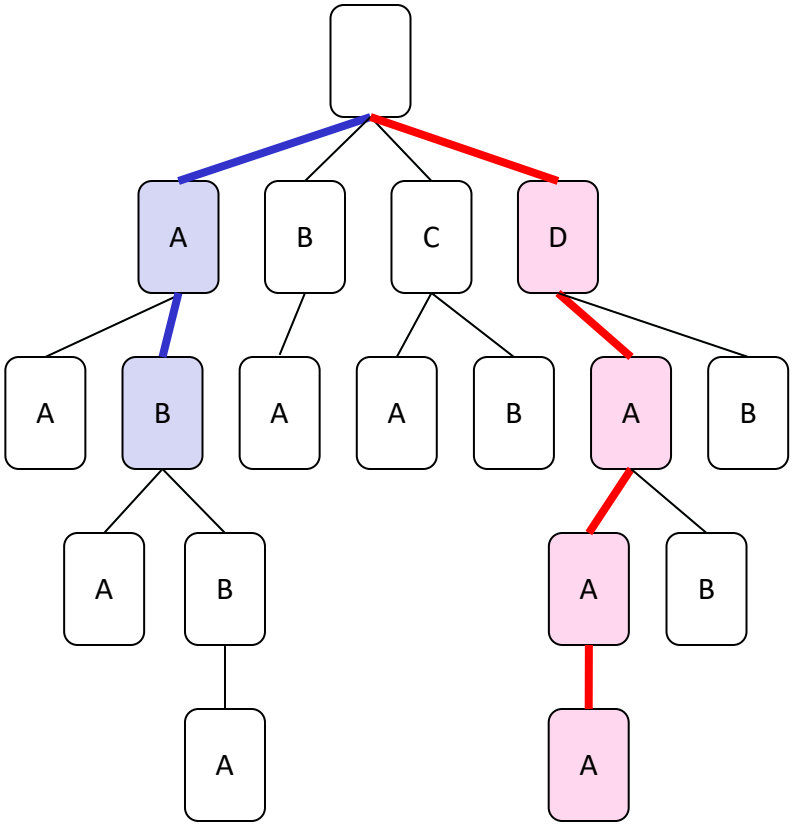
All interior (non-leaf) nodes are directories, shown here in color.

Each directory lists the objects immediately below in the tree. The names of the “child” objects of a given ”parent” must be distinct.

Names can be reused in lower directories.



The leaf nodes are any digital object including directories.



Every node in the tree has a unique path connecting it to the root.

A pathname is the sequence of object names along the path, separated by "/" symbols. Root is denoted by starting the pathname with "/".

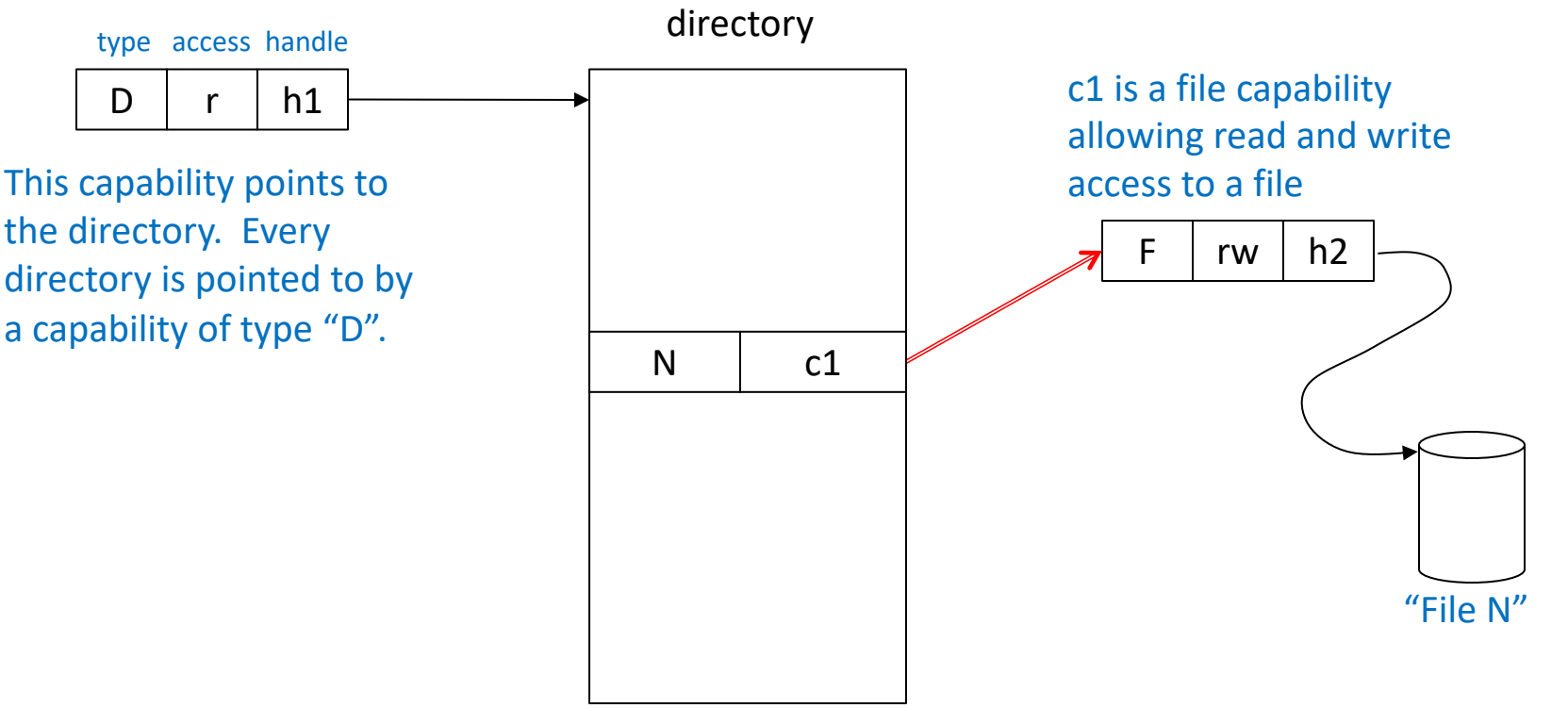
/D/A/A/A is the red path

/A/B is the blue path

An object's pathname uniquely identifies it in the tree and is made entirely of local names chosen by users.

Format of a Directory

- Entries of the form (N,c)
  - N is a symbolic name chosen by the user
  - All symbolic names in directory must be different
  - c is a public capability pointing to an object
- Recall that object type and allowable access rights are encoded into the capability

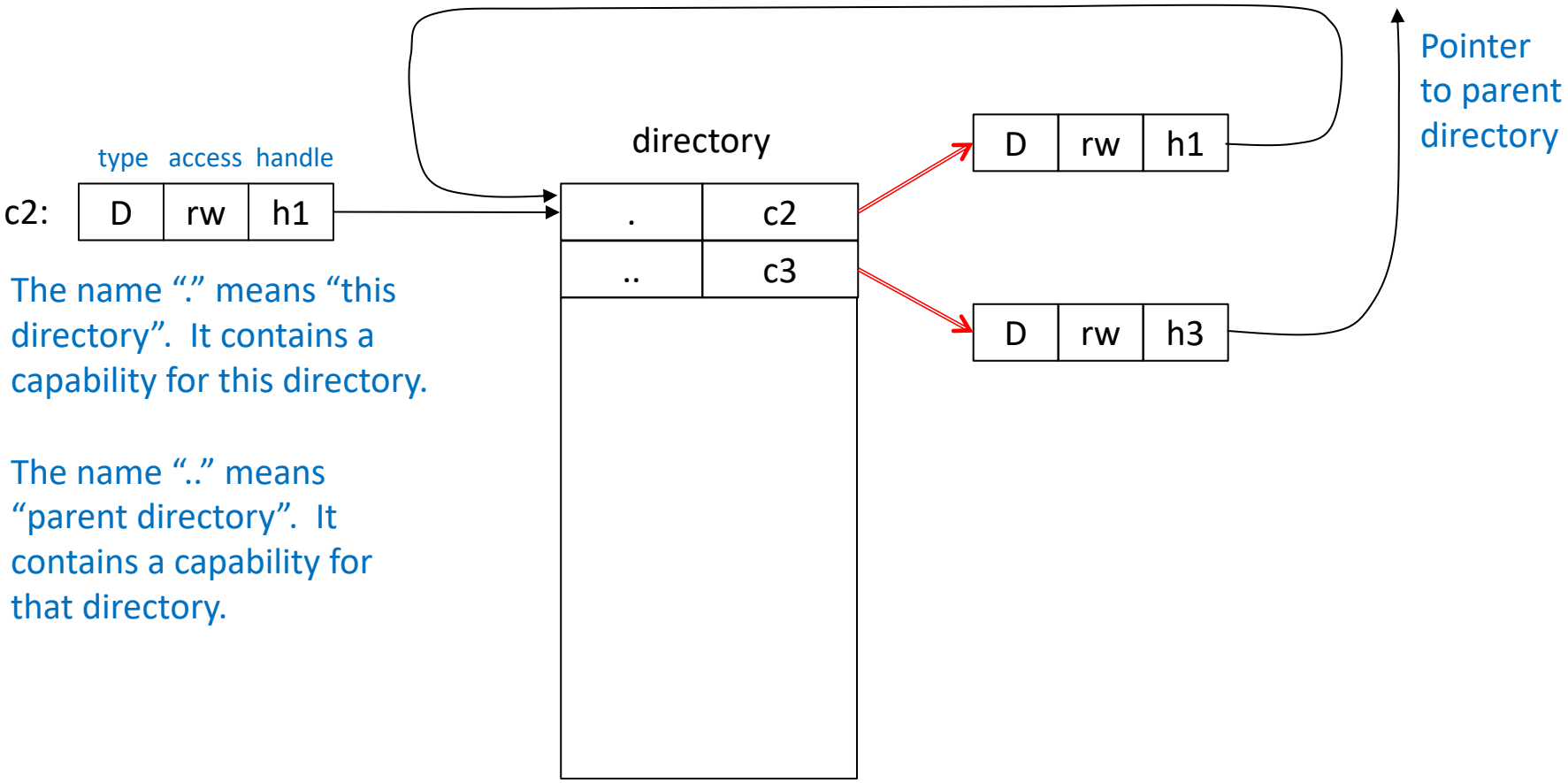


This capability points to the directory. Every directory is pointed to by a capability of type "D".

c1 is a file capability allowing read and write access to a file

Directory is a list of distinct symbolic names paired with capabilities

Directories are managed by kernel. Only public capabilities (with crypto checksums) are stored, thus granting unforgeability.



The name "." means "this directory". It contains a capability for this directory.

The name ".." means "parent directory". It contains a capability for that directory.

UNIX convention: entry "." contains copy of this directory's capability and ".." contains copy of parent director's capability.

Interface to Directory Manager

- TERMINOLOGY
  - "directory CL[i]" abbreviates "directory pointed to by CL[i]"
- i = CREATE_DIR( )
  - Create an empty directory pointed to by directory capability dc, placed in caller's CL[i]
- DELETE_DIR(i)
  - Remove the directory CL[i] and all its contents

Interface - 2

- ATTACH(j,(N,c))
  - Place entry (N,c) in directory CL[j]
  - N must be distinct from other entries already in directory
- REMOVE(j,N)
  - Remove the entry (N,c) from directory CL[j]
- RENAME(j,N,M)
  - Change entry (N,c) to (M,c), provided N exists and M is different from all other names in directory CL[j]

Interface - 3

- c = SEARCH(j,N)
  - Search directory CL[j] for entry with name N, and if found return the capability c associated with it
- c = SEARCH_ON_PATH(pathname,N)
  - pathname = $/D_1/D_2/\ldots/D_n$ (directories)
  - Until i=n, dc_i = SEARCH(dc_{i-1},D_i), where dc_0=root
  - c = SEARCH(dc_n,N)
  - generalize to allow PATH, list-of-pathnames (Unix)

Summary

- Directories map symbolic names to capabilities
- Come in trees
- Every object in tree has unique pathname
- Interface allows
  - Creating and deleting directories
  - adding, removing, and renaming directory entries
  - searching directories and paths for particular name